I have listed my instructions in the order that they should be followed.
I have grouped actions into sections.
You should create a group in the canvas for each section, and use the provided section name as the group label.
You must process each section, and their instructions, in sequential order.
Do not use mock data or mock functions, or return mock results to resolve errors.
Do not create blocks that are identical or whose contents and functions are substantially the same.
Do not create variables, functions, etc. with identical names in different blocks.

Section #1: Data Import
All code for this section should be put into a single box on the canvas.

Import the california housing dataset from scikit.
When you call the fetch_california_housing function, set the data_home parameter to "/tmp/cal_housing".

Set the random seed=1234. This random seed will be used anywhere appropriate in subsequent sections.

Drop any rows from the resulting data set that has a missing target variable.

Section #2: Exploratory Data Analysis
All code for this section should be put into a single box on the canvas.

Print an appropriately annotated box plot for each numeric feature.
Print a statistical summary of the dataset and the first 10 rows of data, along with the feature names.
Calculate the variance of all numeric variables, and covariance between each pair of numeric variables, and print these values as a covariance matrix.

Section #3: Clean Data
All code for this section should be put into a single box on the canvas.

Using scikit, create a pipeline class that will perform the following actions:
1. Impute any missing values with the median value of existing values from the respective column.
2. Standardize all numeric variables such that each variable has a mean of 0 and a standard deviation of 1.

Apply this class to the numeric feature variables of the housing data.

Section #4: Training/Test/Validation Split
All code for this section should be put into a single box on the canvas.

Using scikit, creating a pipeline class that will perform the following actions:
1. Randomly sort the input data.
2. Split the data into three groups: Training, Validation, and Test
3. The Training data set should contain 70% of the input data, Validation 20% of the input data, and Test 10% of the input data.

4. All input data should be present in one, and only one, of these data sets.
5. Each of these data sets has an associated target variable. Split each data set into two subsets,one data set consisting of the feature variables and one data set consisting of the target variable.
The naming scheme of these data sets should be [original name]_features and [original name]_target.
For example, if the input data set name is Training, it will be split into Training_features and Training_target.

Apply this class to the resulting data set.

Section #5: Neural Networks
Use a Fargate compute instance for this block.
The code for this section may be put into a single box on the canvas or multiple boxes, as appropriate

Use the previously-created data to build a neural network that will predict target values using feature values.
For each combination of H in {4,5} and N in {20,21}, requirements for these networks are:
1. H hidden layers
2. Each hidden layer has N nodes.
3. Use the ELU activation function and he_normal kernel initializer in each hidden layer.
4. Use batch normalization after the activation function in each hidden layer.
5. The final output layer should have a single output node and use the ReLu activation function.
6. Use l2 regularization in each hidden layer and the output layer with regularization factor=0.01.
7. Add a dropout layer after each hidden layer with a dropout rate=0.2.
8. Use Adam optimization with beta 1=0.95 and beta 2=0.9995. The learning rate should decrease via exponential decay with initial learning rate=0.01, step size=10, and decay rate=0.1. Other parameters may remain at default settings.

Use the Training_features and Training_target data sets to train the network parameters. Use mean squared error to calculate loss.

At the end of each epoch, evaluate the loss using the Validation_features and Validation_target data sets.
At the end of each epoch, print:
1. The epoch number
2. The latest loss calculated using the Training_features and Training_target data sets
3. The loss calculated using the Validation_features and Validation_target data sets

Training should continue for 50 epochs or until loss calculated using the Validation_features and Validation_target does not lower after 15 consecutive epochs.

Select the model with H and N values that produces the smallest validation loss.

Save this model to variable model_nn and its test loss to variable test_loss_nn.

Finally, evaluate this neural network using the Test_features and Test_target data sets by calculating loss.
Print this loss.

Create a single figure with a multi-panel (e.g., 2x2) layout, where each subplot corresponds to an individual model configuration (e.g., a specific combination of H and N). For each subplot, plot the training loss, validation loss, and test loss (as a horizontal line) as a function of epoch number. Use distinct colors and a legend for each curve, clearly annotate each panel with the configuration values, and add an overall figure title.

Additionally, create a single aggregate plot comparing all H/N combinations using the same loss metrics and axes.

Use the Tensorflow Keras summary() method to print a summary of the model's architecture.

Section #6: Linear Models
All code for this section should be put into a single box on the canvas.

Combine the training and validation feature data into a single data set called linear_train_features.
Combine the training and validation target data into a single data set called linear_train_target.

Build the following linear models using these new linear training features and target data sets.

#1. Build a linear regression model with up-to 2-way interactions. This model should include an intercept term. Calculate the loss of this model when evaluated on the test data.
#2. Build a linear model using LASSO/L1 regularization for each value of alpha in {0.5, 1.5, 2.0, 2.5, 3.0}. Calculate the loss of each model when evaluated on the test data.
#3. Build a linear model using RIDGE/L2 regularization for each value of alpha in {0.5, 1.5, 2.0, 2.5, 3.0}. Calculate the loss of each model when evaluated on the test data.
Print the test loss for each of the models created. Clearly annotate the model number and alpha parameter, if appropriate.

Finally, identify the linear model with the lowest test loss. Save this model to variable model_linear and its test loss to variable test_loss_linear.

Section #7 Final Model Section
All code for this section should be put into a single box on the canvas.

Compare the linear model that had the lowest test loss to the neural network that had the lowest test loss.

Provide a recommendation for which model we should use if we want to minimize loss on test data.